Debugging E3SM Atmosphere Model

A new tool inspired by Perturbation growth test method

Balwinder Singh, Phil Rasch and Hui Wan

“The most secure code is the code that is never written” – Colin Percival
Outline

• Classes of model errors
• Reasons E3SM like codes are harder to debug
• Commonly used debugging tools
• EAM’s Inbuilt debugging tools
• Test cases
Typical Workflow

- **Configure**: Compset, grid etc.
- **Compile**: ./case.build
- **Run**: ./case.submit
- **Analyze Model output**: E3SM Diags
Classes of Model Errors

Configure issues:

```
ERROR: Command: 'components/cam/bld/configure -s -ccsm_seq -ice none -ocn docn -comp_intf mct -dyn se -
dyn_target preqx -res ne4np4 -phys cam5 -clubb_sgs -microphys mg2 -chem linoz mam5 resus mom -
rain_evap_to_coarse_aero -nlev 72 ' failed with error 'tar: Buildconf/camconf/chem_proc/cam.subs.tar:
Cannot open: No such file or directory
tar: Error is not recoverable: exiting now
```

Compile time errors:

```
qfs/people/sing201/eagles/giant_mode/E3SM/components/cam/src/chemistry/modal_aero/dust_model.F90(12):
error #6404: This name does not have a type, and must have an explicit type.  [DUST_NAMES]
public :: dust_names
```
Classes of Model Errors

Runtime errors:

<table>
<thead>
<tr>
<th>1: Image</th>
<th>PC Routine</th>
<th>Line Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: e3sm.exe</td>
<td>000000009A5520A</td>
<td>Unknown</td>
</tr>
<tr>
<td>1: e3sm.exe</td>
<td>shr_abort_mod_mp_</td>
<td>114 shr_abort_mod.F90</td>
</tr>
<tr>
<td>1: e3sm.exe</td>
<td>shr_abort_mod_mp_</td>
<td>61 shr_abort_mod.F90</td>
</tr>
<tr>
<td>1: e3sm.exe</td>
<td>cam_abortutils_mp</td>
<td>59 cam_abortutils.F90</td>
</tr>
<tr>
<td>1: e3sm.exe</td>
<td>rad_constituentsp</td>
<td>1653 rad_constituents.F90</td>
</tr>
</tbody>
</table>

Wrong answers!
Why E3SM is harder to debug?

- Parallel nature of the code
- Size of the code
- Different coding styles

There are two types of people.

Programmers will know.

*E3SM branched off from CESM
Why E3SM is harder to debug?

- Lots of dependencies
- Scattered error log files
- Configurable in many ways

“It works on my machine” -- Anonymous 😊
Common Debugging Tools

Good old print statements

```
Write(*,*) 'Temperature after rad:', state%t(i,k)
```

Debuggers/Compilers

“If you lie to a compiler, it will get its revenge” -- Henry Spencer

Opensource Debuggers

Valgrind

GDB
The GNU Project Debugger
Inbuilt Debugging Tools

Physics Debug Codes:

- Allows to focus on one grid point (lat, lon) on the globe

PERGRO Driven debugging:

- Allows to track how a physical process impacts state variables
- Uses **Physics Debug Codes** to find the offending line of code
Why this tool is very critical?
- Parallel code – Chunks and columns
- Indices of a variable can not be trusted to stay the same
- Allows us to identify a latitude and longitude combination in a model run using chunk
- Namelist changes:
  ```
  phys_debug_lat=67.50000
  phys_debug_lon=28.70000
  ```
- Inside EAM source code:
  ```
  icol = phys_debug_col(chnk_id)
  if(icol>0)write(*,*) 'taux', taux(icol)
  ```
- Proposed Enhancements:
  - Vertical level and Constituent number
PERGRO Test Driven Tool

- Inspired from perturbation growth test
- Stores model output after every physical process
- Helps in tracking which state variable is affected by which physical process

Proposed enhancements:
- Ability to add/remove tracked state variables at runtime (Namelist)
- Ability to track each sub-step of processes taking sub-steps (e.g. CLUBB and MG2)

pergro_test_active = .true.
Scenario – A Broken Restart Test

- Scenario: Modified code to add an enhancement but it broke the model’s BFB restart capability

- First check all the obvious places
  - Carefully review new code modifications
  - Do we need new variables in the restart file?
  - Use a debugger/print statements to review the code

- Last resort – Isolate and understand the code causing non-BFB behavior
How to use these tools?

- Ways to expedite debugging:
  - Reproduce the problem:
    - On a coarsest possible resolution
    - With the least number of time steps (ideally one-time step)
    - Switch off compiler optimization
    - Use all compiler debugging options
  - Use your prior experience with E3SM

- Pergro: Isolate the physical process causing model’s state variables to differ
- Script: Find model’s state variable with max difference
  - Find latitude and longitude of the max difference
- Phys: Isolate line of code causing this difference
- Debug:
Common Test Cases

- Unexpected Non-BFB model results:
  - Broken model restart
  - Perceived BFB code modifications causing answers to change
  - Non-BFB results due to broken threading

- Wrong answers!
  - Value of a variable going out of range or beyond expectation
  - Bugs in the computing environment
Some Recent Debugging Exercises

- Compiler bug (Compy, Intel 19.0.3):
  - MMF and phys_loadbalance

```fortran
  do k = 2, nz-1
    k_wp3 = 2*k - 1
    k_wp2 = 2*k
    rhs(k_wp3) = rhs(k_wp3) + invrs_dt
    rhs(k_wp2) = rhs(k_wp2) + invrs_dt
    rhs(k_wp2) = rhs(k_wp2) - 0.1D0
  enddo
```

- Non-BFB radiation diagnostic code:
  - Processes invoked with different “states”
  - Identified missing processes

- MAM with added inactive mode
  - Identified several answer changing places in code
  - Found a bug in ways sea-salt indices are stored and used in the code